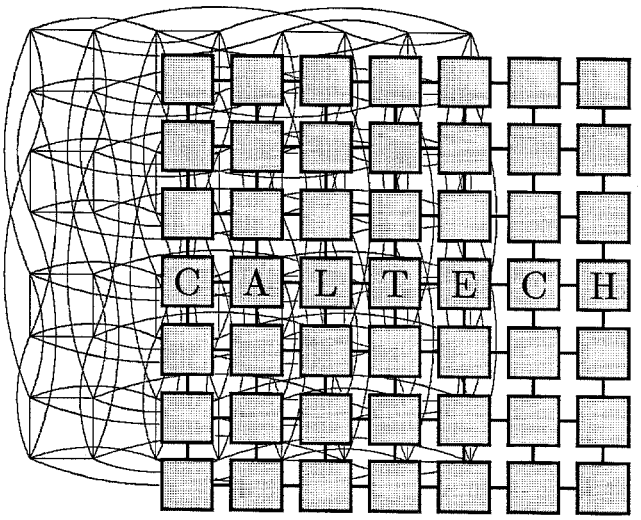
Submitted



SUBMICRON SYSTEMS ARCHITECTURE PROJECT
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125

Semiannual Technical Report

Caltech Computer Science Technical Report

Caltech-CS-TR-88-18

9 November 1988

The research described in this report was sponsored by the Defense Advanced Research Projects Agency, DARPA Order number 6202, and monitored by the Office of Naval Research under contract number N00014-87-K-0745.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
09 NOV 1998		**09-11-1998 to 09-11-1998**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Submicron Systems Architecture Project	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Defense Advanced Research Projects Agency,3701 North Fairfax Drive,Arlington,VA,22203-1714**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**16**	

SUBMICRON SYSTEMS ARCHITECTURE

Semiannual Technical Report

Department of Computer Science
California Institute of Technology

Caltech-CS-TR-88-18

9 November 1988

Reporting Period:	1 April 1988 – 31 October 1988 (7 months)
Principal Investigator:	Charles L. Seitz
Faculty Investigators:	William C. Athas
	K. Mani Chandy
	Alain J. Martin
	Martin Rem
	Charles L. Seitz
	Stephen Taylor

Sponsored by the
Defense Advanced Research Projects Agency
DARPA Order Number 6202

Monitored by the
Office of Naval Research
Contract Number N00014-87-K-0745

SUBMICRON SYSTEMS ARCHITECTURE

Department of Computer Science
California Institute of Technology

1. Overview and Summary

1.1 Scope of this Report

This document is a summary of the research activities and results for the seven-month period, 1 April 1988 to 31 October 1988, under the Defense Advanced Research Project Agency (DARPA) Submicron Systems Architecture Project. Previous semiannual technical reports and other technical reports covering parts of the project in detail are listed following these summaries, and can be ordered from the Caltech Computer Science Library.

1.2 Objectives

The central theme of this research is the architecture and design of VLSI systems appropriate to a microcircuit technology scaled to submicron feature sizes. Our work is focused on VLSI architecture experiments that involve the design, construction, programming, and use of experimental message-passing concurrent computers, and includes related efforts in concurrent computation and VLSI design.

1.3 Changes in Key Personnel

Dr. William C. Athas completed his appointment as a Postdoctoral Research Fellow in Computer Science in August 1988, and has joined the faculty at the University of Texas at Austin as an Assistant Professor of Computer Science. Dr. Stephen Taylor, a new PhD from the Weizmann Institute of Science and the author of a multicomputer implementation of flat concurrent prolog, joined the project in September 1988 with an appointment at Caltech as an Instructor in Computer Science.

2. Architecture Experiments

2.1 Mosaic Project

Bill Athas, Charles Flaig, Glenn Lewis, Jakov Seizovic, Don Speck, Wen-King Su, Tony Wittry, Chuck Seitz

The Mosaic C is an experimental multicomputer with single-chip nodes, currently in development. The stipulation that the nodes fit on a single chip so limits the storage for each node that relatively fine-grain concurrent programming techniques must be used. The Mosaic C will be programmed using the Cantor programming language, a fine-grain object-based (or Actor) language. We are working toward building a 16K-node Mosaic system using nodes fabricated in 1.2μm CMOS technology, with a near-term milestone of a 1K-node system using nodes fabricated in 1.6μm CMOS.

Much of our effort in this period has been concentrated on the Mosaic C project. The following is a brief summary of these activities (See also sections 3.1 & 4.5):

1. Cantor version 2.2 has been used internally within the research group for the past several months, and has been documented for external distribution. A technical report describing a collection of exemplary Cantor 2.2 programs that range up to 15 pages of program text in length was published. The report also reports the rationale for many of the design decisions in the evolution of Cantor from version 2.0 to 2.2.

2. Our initial implementation of a Cantor code generator for the Mosaic C indicated that only a simple procedure call mechanism was required; otherwise, the Mosaic C instruction set has been an efficient target for code generation. Work has commenced on a final Cantor code generator and runtime system for the Mosaic.

3. In accordance with the studies of code generation, the microcode for the Mosaic C processor was revised to implement an instruction set having a simpler procedure-call mechanism, together with several other minor refinements. The simplification of the instruction set reduced the number of implicants in the microcode that controls the processor from 66 to 102. The impact of this simplification on the processor area is merely favorable; its greatest benefit is in improving the processor speed (the RISC effect).

4. The entire processor was simulated at the clock-cycle and microcode level to debug and verify the microcode. The verified microcode was then used to generate a PLA structure, which was tied to the Mosaic C datapath for switch-level simulation and verification of the entire processor. A hybrid static/precharge PLA was designed to maximize the performance, and will be used in the final version of the processor.

5. An interface between the router and memory was designed, laid out, and verified by switch-level simulation. This final section of the Mosaic C single-

chip multicomputer node also includes the arbitration for memory refresh and memory access.

Fabrication of the first prototype processors and full Mosaic elements is now anticipated for early CY1989.

2.2 Second-Generation Medium-Grain Multicomputers*

Chuck Seitz, Alain Martin, Bill Athas, Charles Flaig, Jakov Seizovic, Craig Steele, Wen-King Su

Deliveries of the first production models of the Ametek Series 2010, a second-generation medium-grain multicomputer developed as a joint project between our research project and Ametek Computer Research Division, took place in this period. The reports we have received have been favorable. One customer who is also a DARPA contractor had developed 10,000+ lines of source code using the Cosmic Environment prior to taking delivery of the Ametek 2010, and apparently ported this code in a few days with no difficulties.

Additional benchmarks on the Ametek Series 2010 continue to show that it runs 8–10 times faster per node than such first-generation machines as the Intel iPSC/1.

Copies of the Cosmic Environment system have been distributed to approximately an additional 35 sites in this period, bringing the total copies distributed directly from the project to over 150. In addition, source copies of the Reactive Kernel node operating system were provided to two government contractors who are purchasing Ametek 2010 systems. An article titled "Multicomputers: Message-Passing Concurrent Computers" was published in the August 1988 issue of *IEEE COMPUTER*. This article on the current status of the multicomputers that have developed out of the work of our research group stimulated requests for many additional copies of "The C Programmer's Abbreviated Guide to Multicomputer Programming" [Caltech-CS-TR-88-1].

We expect to take delivery of the first 16-node increment of a 256-node Ametek 2010 in November 1988, and also a 16-node Intel iPSC/2, which will later be expanded to 64 nodes. Substantial blocks of time on the Ametek 2010 will be available to guest DARPA researchers.

Our Caltech project continues to work with both Ametek and Intel on the architectural design, message-routing methods and chips, and system software (evolutions of the Reactive Kernel (RK) node operating system and the Cosmic Environment (CE) host runtime system) for multicomputers. (See sections 3.2, 3.6 and 4.6 for details on these efforts.) We expect to see additional major advances in the performance and programmability of these systems over the next two years. In

* This segment of our research is sponsored jointly by DARPA and by grants from Intel Scientific Computers (Beaverton, Oregon) and Ametek Computer Research Division (Monrovia, California).

addition, we continue to develop applications in VLSI design and analysis tools, and in other areas in which the programming of these multicomputer systems presents particular difficulties or opportunities. (See sections 3.3–3.5 and 4.9.)

2.3 Cosmic Cube Project

Bill Athas, Wen-King Su, Jakov Seizovic, Chuck Seitz

This section summarizes the current usage and the hardware and software status of our first-generation multicomputers, the Cosmic Cubes and Intel iPSC/1 d7.

These systems continue to operate reliably. Overall usage has been moderately heavy. The most time-consuming application in this period from within our own group has been a continuation of an extensive series of simulations by John Ngai concerned with the maximal utilization of networks with faulty routers or channels (see section ?). Supersonic flow computations being performed by students and faculty in Aeronautics at Caltech continue as the largest share of outside use.

The 64-node Cosmic Cube exhibited a hard failure in this seven-month period, a complete failure of its primary 5V, 130A power supply. The power supply was replaced, and the system rebooted without any problems. Counting the power supply failure as a single failure, the two original Cosmic Cubes have now logged 3.6 million node-hours with only four hard failures, three of them being chip failures in nodes. Curiously, we have not encountered a single connector failure. The calculated node MTBF of 100,000 hours reported before these machines were constructed was extremely conservative. A node MTBF in excess of 1,000,000 hours is probable, and can be stated at a 54% confidence level.

Our Intel iPSC/1 d7 (128 nodes) was contributed to the Submicron Systems Architecture Project as a part of the license agreement between the Caltech and Intel, and is accessible via the ARPAnet to other DARPA researchers who may wish to experiment with it. To request an account, please contact `chuck@vlsi.caltech.edu`. The Ametek Series 2010 system to be installed later this month will be available for outside use on a similar basis.

3. Concurrent Computation

3.1 Cantor

Nanette J. Boden, William C. Athas, Chuck Seitz

Programming for Fine-Grain Multicomputers

Over the last year we have been conducting a series of fine-grain programming experiments using Cantor. The purpose of this series of experiments was both to evaluate Cantor as a programming language and to investigate the nature of fine-grain programming. Application programs that have been written in these experiments include: fast-Fourier transform, shortest-path algorithms, a 2D convex hull solver, R-C chain-circuit simulation, digital logic simulation, a checkmate analyzer, an enumerator of paraffin isomers, and many others.

As a result of these programming experiments, modifications to Cantor have been made to facilitate fine-grain programming. Iteration internal to objects, custom objects, functional abstraction, and one-dimensional vectors are programming constructs that are now available in the newest version of Cantor, Cantor 2.2. A feature has also been added to the language to permit rudimentary discretion over message receipts. Analysis of the programming experiments clearly indicates that programming situations exist where some message discretion is very useful. In addition to these modifications, unnecessary features of the original language specification have been removed, including dynamic typing of variables. The changes that have been made to Cantor thus enhance programming abstraction while removing unnecessary constructs.

Using the latest version of Cantor as an experimental tool, we have written enough programs in the fine-grain style to draw some conclusions. Although formulations for Cantor programs are myriad, we have detected three general paradigms for the development of fine-grain programs:

1. Functional program specifications can be mapped directly into message-driven programs.
2. Solution specifications can be mapped into message-driven programs.
3. The object program can operate as a "logical apparatus" to solve the application problem.

In addition to observing these paradigms, we have been encouraged by the high degree of concurrency that is achieved in Cantor programs and by the convenience and generality of fine-grain programming. Based on our experiments with Cantor thus far, we believe that large, highly concurrent programs can be efficiently expressed in the fine-grain programming style.

Programming for the Mosaic

Recent research in the area of Mosaic programming has focused on the definition and analysis of an abstract machine for the execution of Cantor code. The Cantor Abstract Machine (CAM) definition is based on the fine-grain multicomputer architecture, yet encapsulates operations like object creation, message sends and receives, *etc,* in single instructions. The purpose of this approach is to isolate the implementation of these complicated operations as much as possible from the development of an efficient runtime system.

A new Cantor code generator and simulator have been written for the CAM. Analysis of the abstract machine has already suggested improvements in the Cantor intermediate format. In addition, simulation of program execution on the CAM is expected to be very useful in evaluating potential Mosaic runtime system alternatives.

3.2 The Cosmic Environment and Reactive Kernel

Jakov Seizovic, Wen-King Su, Chuck Seitz

The Cosmic Environment and Reactive Kernel continue to run reliably on the original Cosmic Cubes and on the Ametek Series 2010, and no major changes have been made. The internals of RK are now documented in technical report Caltech-CS-TR-88-10.

In the original version of the RK, we were able to guarantee the *weak fairness* of scheduling on a multicomputer node only if all processes on that node satisfied the reactive property that they would eventually either terminate, or execute an `xrecv()`. The producers of an infinite number of messages are an important class of processes that do not satisfy the reactive property. A simple modification of the implementation of the `xmalloc()` system call has enabled us to support the infinite computations as well. The `xmalloc()` system call is implemented in terms of the RPC mechanism. The requested buffer is not delivered immediately; instead it is sent to the requesting process and delivered through the regular scheduling mechanism.

3.3 CONCISE — A Concurrent Circuit Simulator*

Sven Mattisson, Lena Peterson, Chuck Seitz

Within this project, a concurrent circuit simulation program called CONCISE has been developed. This program is a circuit simulator for transient analysis of CMOS-circuits. It is written in C and uses the Cosmic Environment/Reactive Kernel message-passing primitives.

* This segment of our research is a joint project between the Caltech Submicron Systems Architecture Project and the Department of Applied Electronics at the University of Lund, Sweden.

Recently, CONCISE was ported to the Ametek Series 2010. Thus, the program now runs on several multicomputers with loosely coupled nodes, including the Ametek 2010 and the Intel iPSC, and on a shared memory multicomputer, the Sequent Symmetry. The port to the Ametek 2010 showed that CONCISE is more than eight times faster on the Ametek 2010 than on the Intel iPSC/1, which is a typical first-generation multicomputer.

The Reactive Kernel primitives support a programming model where each process has its own memory space. This model makes dynamic partitioning and load balancing expensive in CPU time. Thus, we have developed a static partitioning scheme that tries to enhance the convergence rate of the waveform relaxation method without sacrificing the grain-size of the computational tasks. It is important to notice that the requirements on the partitioning algorithms in this case differ from the "traditional" parallelization, where only a few processing nodes are used.

So far, six different combinations of iteration schemes and partitioning have been tested. The iteration schemes tested are ordinary Jacobi iterations, ordinary Gauss-Seidel, and n-colored Gauss-Seidel. The n-colored Gauss-Seidel uses the incidence-degree algorithm to find a coloring with the least number of colors for the circuit graph. Then, the different colors can be solved concurrently, since each node has a color different from those of its neighbors. These three algorithms have all been run with two different partitioning schemes: one in which each circuit node forms a cluster on its own, and one where source-drain connected circuit nodes are clustered together.

The results show that regular Gauss-Seidel iterations are not suitable except for very few processing nodes, and this scheme is the most popular for sequential waveform-relaxation implementations. Instead, the n-coloring version of Gauss-Seidel iterations are useful for the case when the number of processing nodes is large, but significantly less than the number of processes. The number of colors needed usually lies between three and five.

When the number of computing nodes is close to the number of circuit nodes, Jacobi iterations do surprisingly well. This is due to the fact that the load imbalance gets increasingly severe for the other schemes. For some circuits, the clusters get very big, and splitting schemes fail in producing reasonable size clusters that still achieve comparable convergence speed. For such circuits a hierarchical approach where more than one node can be assigned to solving a cluster would be desirable. Such an approach will be possible with the faster message passing of the second-generation multicomputers, and experiments in this area are presently being carried out.

In another effort, Concise has been used by Anthony Skjellum in the Chemical Engineering Department at Caltech for the simulation of distillation columns. This work has shown that it is possible to use Concise to simulate dynamic systems that

are not at all like circuits. As part of this effort, Concise has been modified to make it easier to install models of other kinds of "devices."

3.4 Variants of the Chandy-Misra-Bryant Distributed Discrete-Event Simulation Algorithm

Wen-King Su, Chuck Seitz

A new and more versatile logic simulator has been written in the past six months to better evaluate a more diverse set of conservative variants of the Chandy-Misra-Bryant (CMB) distributed discrete-event simulation algorithm. Most of the conclusions from this study are included in the paper "Variants of the Chandy-Misra-Bryant Distributed Discrete-event Simulation Algorithm," accepted for publication in the 1989 SCS Eastern Multi-conference. The primary conclusions are that the variants examined are similar, in that all of them take an initial penalty running on a single node in comparison with sequential event-driven simulators that exploit an ordered event list. The penalty is due to the generation and the processing of null messages. However, as the number of processing nodes increases, the simulation time decreases linearly until all usable concurrency has been exhausted. Depending on the circuit being simulated, the crossover point (the point at which the time taken by the concurrent simulators drops below the time taken for the sequential simulator) has been observed to be anywhere between four and 200 nodes.

After the paper was submitted, a new simulator variant was written to try to reduce the initial overhead by combining sequential simulation methods with the concurrent simulator variants. The resulting simulator has the performace of a sequential simulator for the single processor case, and it converges with that of the concurrent simulator when the number of nodes is sufficiently large. However, the nature of the logic circuit being simulated strongly influences the rate of convergence. We have observed all three cases:

1. The simulation time humps upward toward that of the concurrent simulators as soon as the number of processing nodes is increased beyond one.
2. The simulation time remains the same until the concurrent-sequential crossover point.
3. The simulation time starts to decrease as soon as the number of nodes are increased, but the drop is less than linear.

A conclusion of this study is that very-high-performance logic simulation on concurrent computers is completely plausible for systems with very large numbers of nodes, where the CMB null-message scheme is fully exploited. Conversely, it is efficient for small-N systems only when the elements being simulated are more complex and have longer running times than logic elements.

3.5 Automatic Mapping of Processes and Channels

Drazen Borkovic, Alain Martin

To facilitate programming of message-passing machines, we have developed a preprocessor, **map**^{3}, that allows for a certain level of abstraction in the mapping of processes and channels on the nodes and physical channels of a message-passing multicomputer.

The description of a set of processes and the channels between them has been compiled into a set of C functions that perform the mapping of the processes onto physical nodes of the target machine. The preprocessor supports a hierarchical organization of processes and local names for the channels. There is also a set of library routines that can emulate channels with arbitrary slack.

The preprocessor and the library routines have been successfully implemented and tested under the Cosmic Environment/Reactive Kernel system.

3.6 A Multicomputer "Page Kernel"

Craig S. Steele, Chuck Seitz

As described in a previous report, an experimental "page kernel" is being developed that uses memory-access-protection mechanisms as the interface to multicomputer message subsystems. A prototype of the "page kernel" is now running on a sequential machine. The current code is simulating the memory-management hardware of the Ametek Series 2010 computing node, and will be ported to the Series 2010 shortly.

The page kernel supports dynamic load-balancing and process relocation. The kernel's ability to transparently update copies of data distributed across a multi-node system is particularly well-suited for chaotic iterative programs, such as process-placement optimization.

4. VLSI Design

4.1 Testing Self-Timed Circuits

Pieter Hazewindus, Alain Martin

We are investigating methods to test self-timed circuits. Traditionally, it is thought that these circuits are hard to test because of the possibility of races and hazards, and because these circuits are sequential. In our design method, however, races and hazards are absent.

The fault model we use is the stuck-at model, where each wire may be stuck forever at a high (logic-1) or low (logic-0) voltage. We have proven that it is sufficient to perform a single four-phase handshake on each channel to detect all detectable stuck-at faults. Some faults are undetectable.

For the automatic compilation, the main sequencing element is the so-called D-element. For the D-element, there are twenty-two possible stuck-at faults, two of which are undetectable. We have designed an alternate D-element that does not have any undetectable stuck-at faults. Most other circuit constructs in this compiler are completely testable.

Although it is not yet certain whether all constructs can be made entirely testable, our present estimate is that self-timed circuits designed according to our method should be easier to test than traditional clocked circuits.

4.2 A Self-Timed $3x + 1$ Engine

Tony Lee, Alain Martin

We have designed and fabricated a self-timed special-purpose processor for implementing the $3x+1$ algorithm. The processor consists of a state-machine and an 80-bit-wide datapath. It contains approximately 40,000 transistors and operates at over 8 MIPS in 2μm MOSIS SCMOS technology. As usual, the chip was functional on first silicon.

4.3 Performance Analysis of Self-Timed Circuits

Steve Burns, Alain Martin

We have developed methods for determining the repetition time of a set of communicating sequential processes described as handshaking expansions. This performance measure is provided in the form of constraint equations involving symbolic values of the communication and sequencing delays. The analysis is valid regardless of the actual delay values, and thus provides a means of comparing designs described at the handshaking expansion level without first generating detailed circuit implementations. Circuits for handshaking expansions that result in slow repetition times need never be designed.

This method has proven particularly useful in the analysis of programs involving data. It has been used throughout the design of the self-timed microprocessor, increasing the performance of programs involving data up to a factor of two.

4.4 The Design of a Self-Timed Microprocessor

Alain Martin, Steve Burns, Tony Lee, Drazen Borkovic, Pieter Hazewindus

In order to refute the claims that our design method would be too slow and too wasteful in area for anything but small circuits, we have embarked on the design of complete general-purpose microprocessor. The instruction set is "classic": 16-bit instructions with offset, load/store type of instructions, and separate memories for instructions and data. The only restriction is the absence of an interrupt mechanism.

As expected, since the method is based on concurrent programming techniques, the design is highly concurrent. The fetch, decode, and execute phases overlap, as do the execution of ALU and memory instructions. The different processes share 16 general-purpose registers, and four buses are used to communicate with the registers, in addition to point-to-point channels.

We are now in the layout phase of the design. Preliminary estimates of the performance are encouraging. In 2μm SCMOS, we expect to reach 20MIPS.

4.5 Mosaic Elements

Chuck Seitz, Bill Athas, Charles Flaig, Glenn Lewis, Don Speck, Jakov Seizovic, Wen-King Su

With the completion of the packet interface section and the near-completion of the processor, and with the other sections having already been fabricated and tested, the Mosaic C single-chip multicomputer node is rapidly approaching completion. Assembly of the sections will start within the next month, and fabrication of complete elements early in 1989.

The packet interface for the Mosaic chip has been layed out and verified with the switch-level simulation. It is entirely synchronous, and was designed conservatively, so no problems with it are anticipated.

The packet interface consists of two independent finite-state machines, one for sending packets, and the other for receiving packets. Both machines act as simple DMA channels, stealing unused memory cycles, and the packet interface is designed to be able to sustain a throughput equal to the maximum possible message rate that can be achieved by the message router.

The packet interface provides for a fairly complete testing of itself and the router, initiated by a CPU request to send a message to itself. In this mode of operation, the message will be taken from the memory, sent through all three router dimensions, and received back into the memory.

4.6 Fast Self-Timed Mesh Routing Chips

Charles Flaig, Chuck Seitz

A new design of a mesh routing chip (MRC), the FMRC2.0 design, was sent to fabrication in May 1988, together with a separate test chip containing only the FIFO used in the FMRC2.0. These chips employ a circuit design style that is potentially faster but less conservative than is usual for self-timed designs. The chips returned from fabrication do indeed operate nearly three times faster than previous designs. The FIFO test chip, fabricated in a 2μm MOSIS SCMOS process (this chip was also a test of the new 40-pin 2μm pads and design frame that we developed for MOSIS) operated correctly at 70 MBytes/s!

The critical path in a routing chip includes somewhat longer delay paths due to the switching of the packets; hence, although the FMRC2.0 was fabricated in a 1.6μm process, and its FIFOs might be expected to operate at around 85 MBytes/s, it operates as anticipated at 70 MBytes/s. However, it routes packets incorrectly, showing symptoms of directing packets according to the tail of the previous packet rather than the head of the current packet. This fault was finally traced to a timing error of approximately 0.7ns in the latching of a routing decision. The timing error was fixed, and the timing margins in the entire chip were reexamined. A *post facto* Spice simulation of what the analysis showed were the critical points in the old and new designs verified that the original design had a timing error of 0.7ns, while the revised design has a timing margin of about 1.0ns (about 50% of the difference between two short delay paths; hence, not as close as it may sound).

If successful, we expect this new FMRC chip to replace the MRC currently used in the Ametek Series 2010 multicomputer. With help from George Lewicki, this design is also being transferred to an Intel fabrication process for possible use in a future Intel multicomputer.

Tests of the self-timed FIFO in a 2μm MOSIS SCMOS technology will be of interest to other chip designers in the DARPA VLSI community — particularly those designing self-timed chips.

The 2μm FIFO tests yielded a request $\rightarrow$ acknowledge time of 6.5-7.0ns, and a throughput of over 70 MBytes/s on these byte-wide channels. Lest someone interpret this test result as implying that we are driving 70MHz signals through these pads, please understand that in 2-cycle R/A signaling (*cf,* Mead & Conway, figure 7.16), only one *transition* is required for each data transfer, so the maximum fundamental frequency on any R/A or data pin is 35MHz to transfer data at a 70MHz rate.

The total fall-through time for all 101 FIFO stages was measured as 350ns, or 3.5ns fallthrough per stage. The fallthrough time calculated by the τ-model is about 70τ, so this is consistent with a value of τ for the 2μm MOSIS SCMOS n-well process of about 50ps (which is a bit smaller than expected). The *internal*

cycle time when the operation is not impeded by signals passing through pads and package pins is about 180τ, or about 9ns, corresponding to an internal throughput rate of 114MHz.

These speeds in the 2μ MOSIS n-well SCMOS technology are, as expected, about twice as fast as a nearly identical test device fabricated in a 3μm MOSIS p-well SCMOS process. The fallthrough times are more difficult to measure in the 1.6μm FMRC2.0 chip, because of switching and address-decrementing logic in the FIFO pipeline. We can infer than the FIFO fall-through times are about 2.8ns per stage, corresponding to a τ of 40ps, and an internal throughput rate of about 140 MHz.

It is quite evident from these tests that we are able to achieve much higher internal speeds with self-timed and/or asynchronous designs than we know how to achieve with clocked designs.

4.7 Adaptive Routing in Multicomputer Networks

John Y. Ngai, Chuck Seitz

Our studies of adaptive routing in multicomputer networks are approaching a conclusion, and have been generally successful. We now believe that the Adaptive Cut-Through (ACT) routing scheme is capable of outperforming the existing highly evolved oblivious routing devices by a factor of about two in throughput, and have numerous other advantages in hot-spot throughput and fault-tolerance. A summary of the results of our investigations is attached at the end of this report.

What remains to be done to realize the advantages of the ACT routing scheme is to design a VLSI routing chip and/or a new routing section for the Mosaic C.

4.8 Pads and Pad Frame Generation

Charles Flaig, Chuck Seitz

Derived in large part from the pads and pad frames we have designed for mesh routing chips (MRCs), a variety of new pad circuits have been designed for the $\lambda = 0.6\mu$m, 0.8μm, and 1.0μm MOSIS SCMOS processes. One of these design variations was used to produce a new 2μm 40-pin "tiny-chip" frame for MOSIS, including input, Schmitt input, output, and tristate output pads. The unusual features of these pad designs include the use of longitudinal (bipolar) clamp transistors for static and overvoltage protection, and a variety of pad pitches.

We can now report some test results for the 2μm pads. This 40-pin pad frame was fabricated with a 101-stage self-timed FIFO from the FRMC2.0 design (see section 4.6), together with some output pads being driven directly from input pads.

Overvoltage clamping on the inputs clamps to 6V at 200mA, and 7V at 800mA, which is excellent. Undervoltage protection is about the same as above, BUT, at

about -500mA the chip appears to suffer latchup (if power is supplied). This is not a problem for normal static, where no Vdd is applied, but if an input does goes more than about 1V negative while power is applied, latchup may be induced.

For the Schmitt input pad, trigger voltages are 0.8V and 3.9V, for a 2.9V hysteresis. Inpad → Outpad delay is 1.5–2.0ns for no load, 2.0–2.5ns for a fanout of 1, and 2.5-3.5ns for a fan-out of 2. Rise/fall time is 3.5ns for no load, 4.5ns for a fanout of 1, and 6.5ns for a fan-out of 2. The output pads can sink about 30mA at 1.0V, or source about 30mA at 4.0V, under 5.0V operation. These characteristics are more than adequate for student projects.

4.9 The Notorious CIF-flogger Program

Glenn Lewis, Chuck Seitz

The CIF-flogger is a multicomputer program for flattening CIF files, rasterizing the geometry, and for performing parallel operations on the geometry in strips. It runs under the CE/RK system, and hence, on most available multicomputers, including the Ametek Series 2010.

The CIF-flogger currently supports simple bloat, shrink, and logical operations on the flattened geometry, and hence can perform most geometrical design-rule checks. It establishes connected component labeling and will eventually provide complete design-rule checking, well checks, and circuit extraction. Based on timings on the iPSC/1, CIF-flogger is expected to perform design rule checks for 100K-transistor chips in much less than 1s per rule on second-generation multicomputers.